

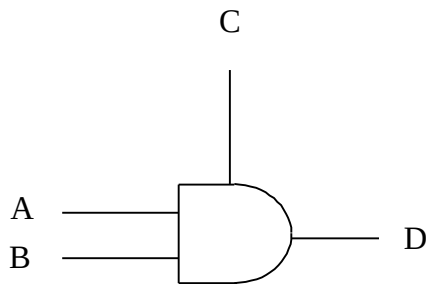
The GAL22V10C: An Introduction

Prepared by: P. David Fisher and Diane T. Rover

This document has the following purposes:

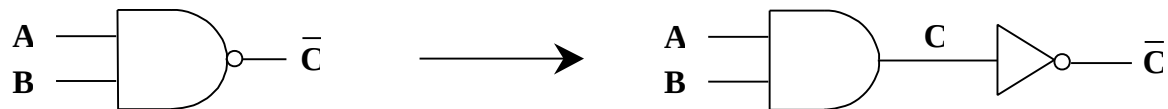
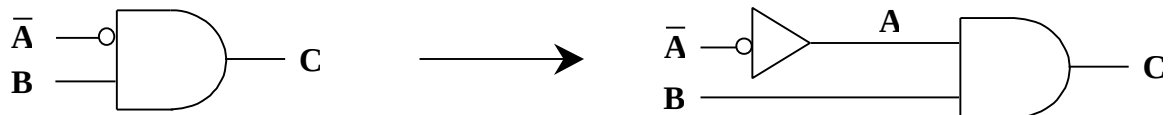
1. It reviews some important digital-logic topics previously learned including:
 - 1.1. Tri-state buffers
 - 1.2. Polarity conventions and terminology
 - 1.3. Graphical symbols
 - 1.4. Flip flops
 - 1.5. Selecting input and output-pin polarities
 - 1.6. Programmable logic arrays (PLAs)
 - 1.7. State-machine models
2. It provides an introduction to the specifications and utility of the GAL22V10C.
3. It provides an introduction to the tools needed to program the GAL22V10C, including:
 - 3.1. The CUPL language and compiler
 - 3.2. The CSIM simulator and debugger
4. It demonstrates how:
 - 4.1. Combinatorial logic designs can be implemented using the GAL22V10C
 - 4.2. Sequential state-machine designs can be implemented using the GAL22V10C

1. Tri-state buffers:



Truth Table			
A	B	C	D
X	X	0	Z
0	0	1	0
0	1	1	0
1	0	1	0
1	1	1	1

2. Polarity Conventions:



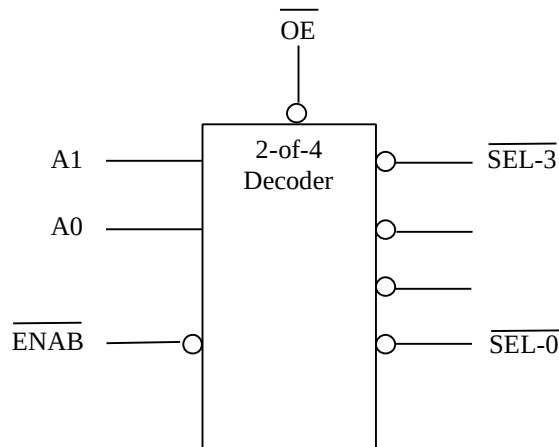
$1 = \text{H (High Voltage)}$
 $0 = \text{L (Low Voltage)}$

$\left. \begin{array}{l} 1 = \text{H (High Voltage)} \\ 0 = \text{L (Low Voltage)} \end{array} \right\} \text{Positive-Logic Convention}$

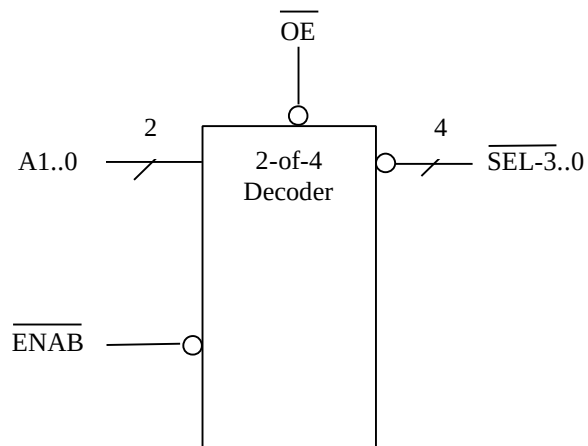
“Assert” may be 1 (H) or 0 (L)

“Not Asserted” may be 1 (H) or 0 (L)

3. Example: A 2-of-4 Decoder

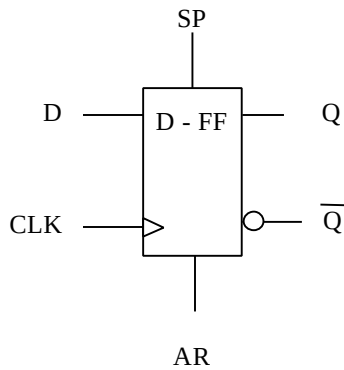


- A. Output Enable is “asserted” low
- B. The decoder is “enabled” by being asserted low
- C. The outputs are asserted low
- D. The tow-bit address lines use “positive logic.”
- E. The above logic function can be drawn as follows:



- F. This decoder is said to be a “combinatorial” logic element.

4. The D- type flip flop (latch).

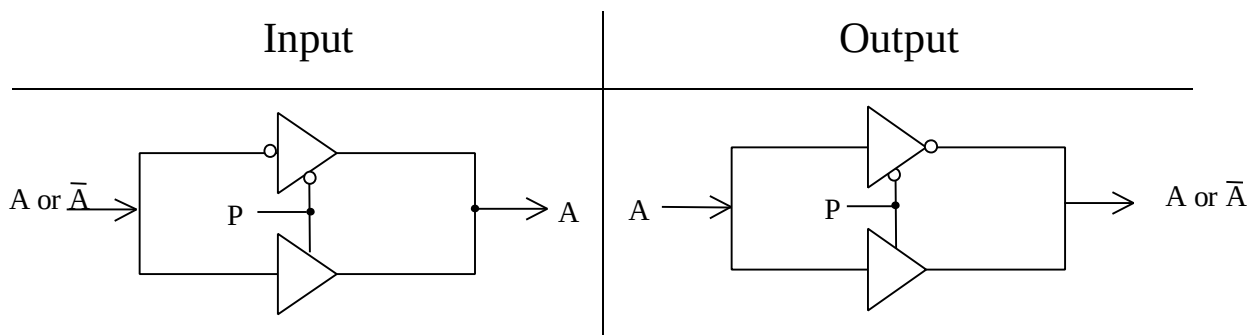


Transition Table

Present Input	Next Output
D_n	$Q_{n+1} = D_n$

- B. The flip flop is positive-edge triggered
- C. The flip flop has an “asynchronous reset” (AR) which is positively asserted.
- D. The flip flop has a “synchronous preset” (SP) which is positively asserted.

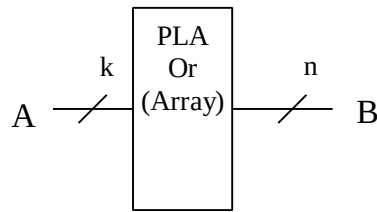
5. Selecting Input and Output Polarities



If $P=0$, $\bar{A}$ is selected.

If $P=1$, A is selected.

6. The Programmable Logic Array (PLA)



$$A = A0..k-1$$

$$B = B0..n-1$$

A is the input data word (k independent variables.) B is the output data word (n independent variables.)

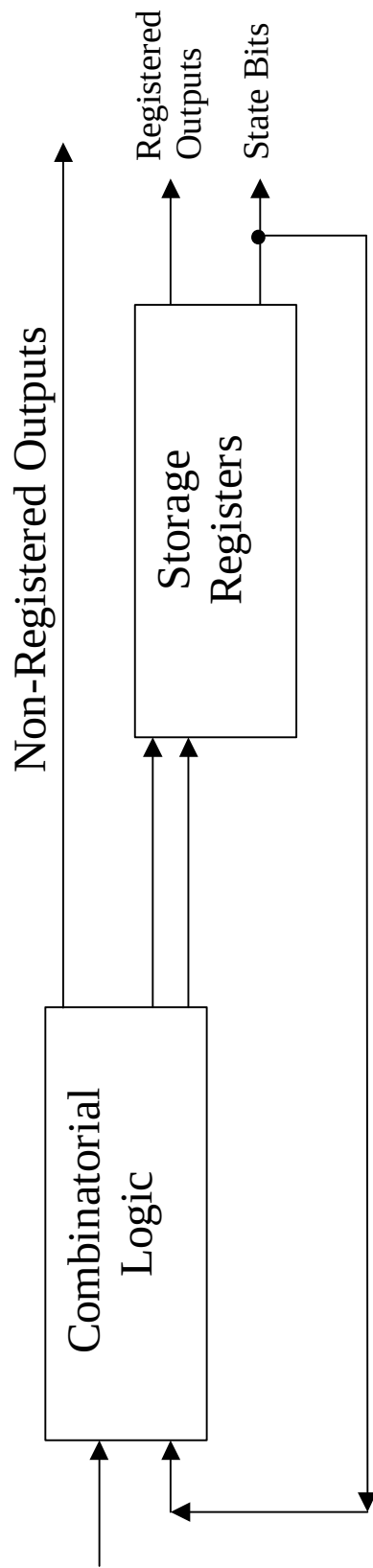
$B = f(A)$ in Sum-of-Products form.

Examples:

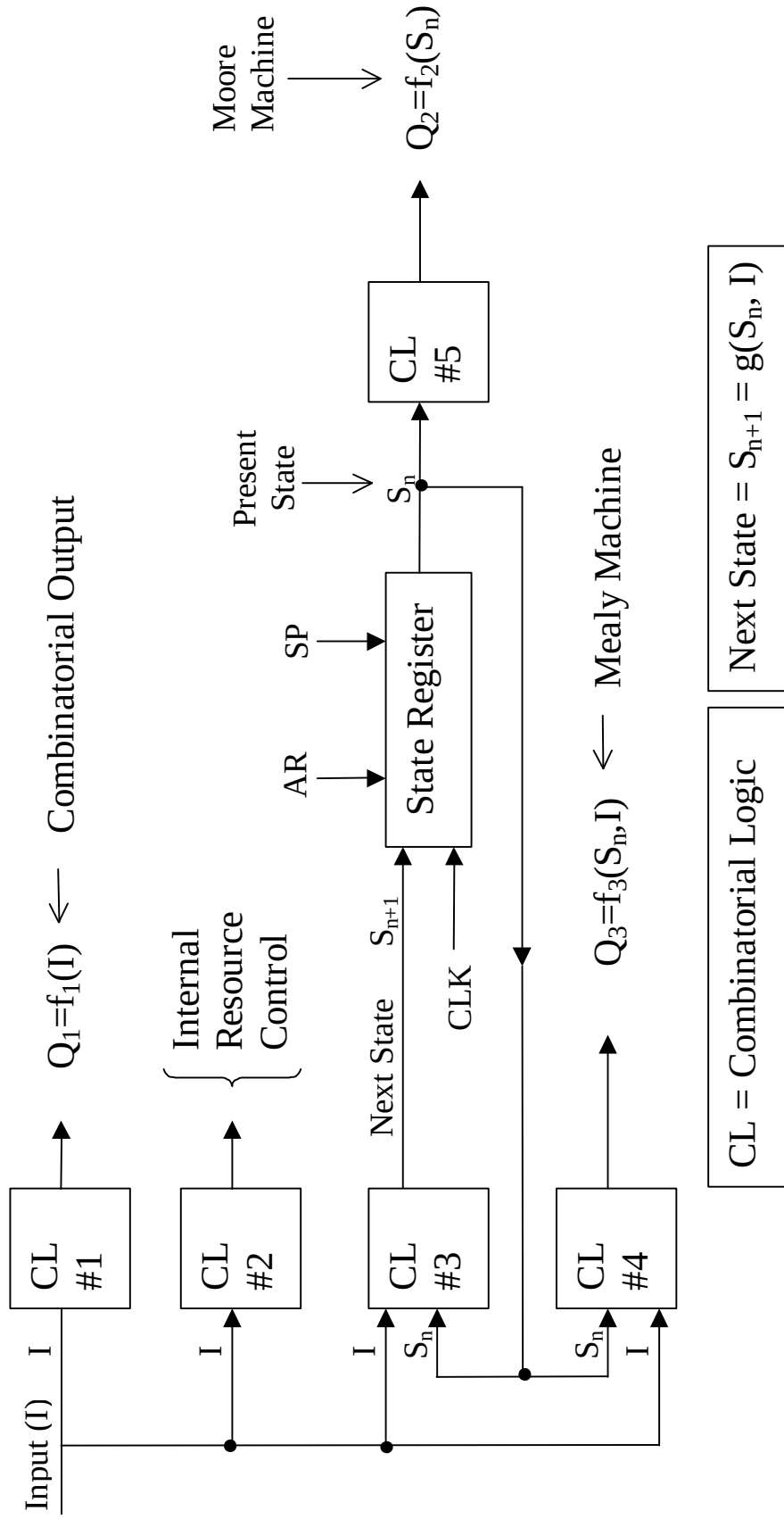
$$\begin{aligned} B0 &= A1'A0' + A3'A1A0 \\ B1 &= A3 \\ B2 &= A1'A0 + A1A0' \\ B3 &= A1 + A2 + A4 + A6 \end{aligned}$$

We often say that the PLA is composed of two planes: the AND plane and the OR plane. The AND plane forms the “product terms;” the OR plane forms the “sum-of-products.”

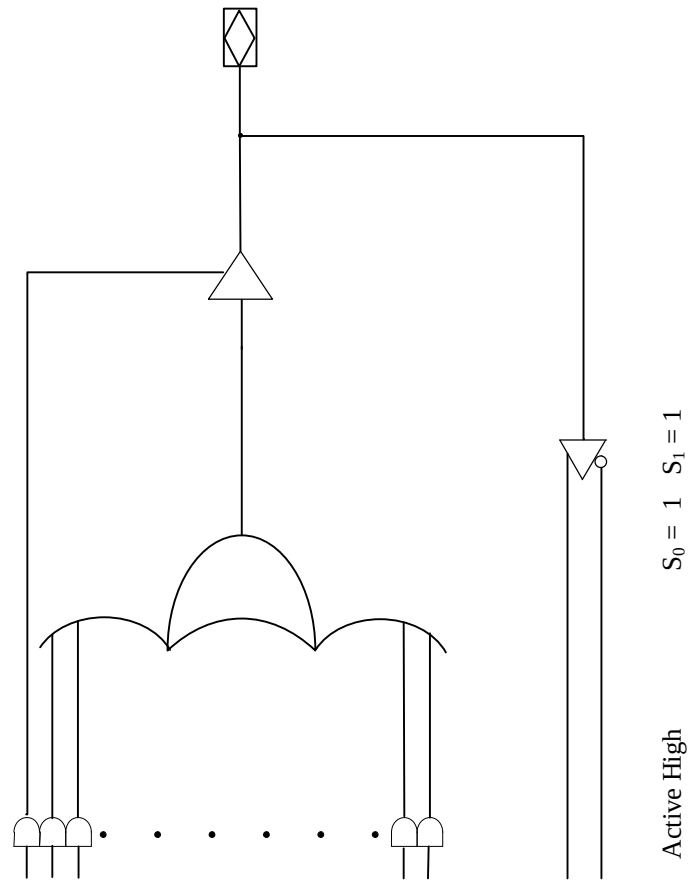
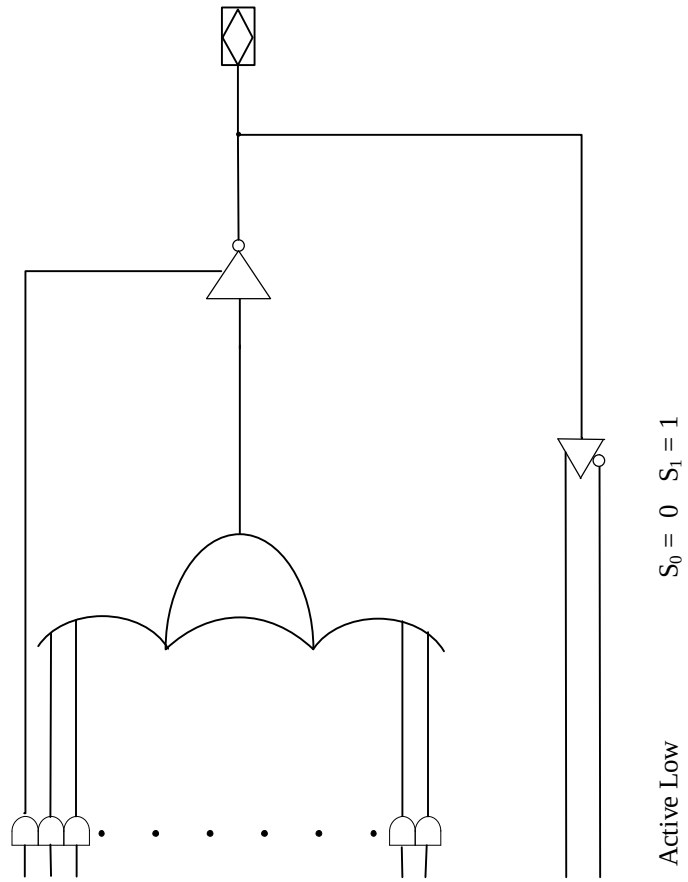
7. Logical Device's State – Machine Model (See Page 215 in CUPL Manual.)



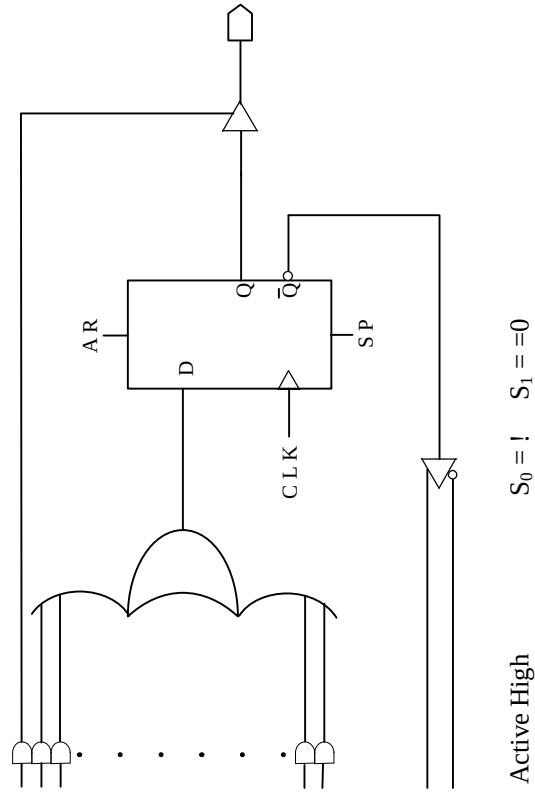
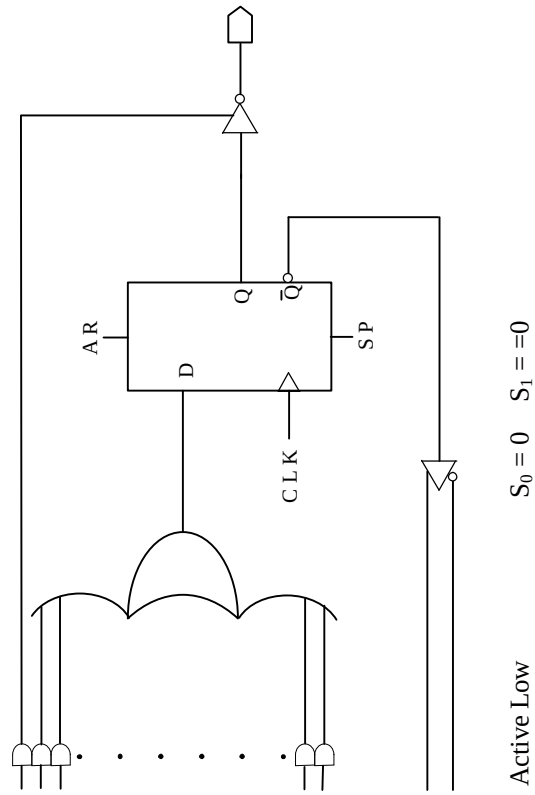
8. Fisher-Rover State-Machine Model (Tuned for Generic Array Logic – GAL) Note: Input Buffers and Output Buffers are not shown.



Combinatorial Mode



Registered Mode



1.Coin-Sorter Demo

Functional Requirements

A combinatorial logic circuit must be designed and implemented using a GAL22V10C to meet the following functional specifications:

1. There are five input pins (d, n, oe2, !oe1, and oe0) and three output pins (Q2, Q1, and Q0).
2. The “d” input represents a dime.
3. The “n” input represents a nickel.
4. The remaining three inputs (oe2, !oe1, and oe0) control the tri-state logic for outputs Q2, Q1, and Q0, respectively.
5. Q2 is asserted if a nickel and a dime are present.
6. Q1 is asserted if a nickel or a dime are present.
7. Q0 is asserted if a nickel is present but not a dime.

```

Name          Demo for a GAL22V10C (DIP);
Partno        Demo_01;
Revision      01;
Date          7/14/98;
Designer      P. David Fisher;
Company       Spartan Embedded Technologies @ Michigan State University;
Assembly      Prototype;
Location      3230 EB;
Device        GAL22V10C (DIP);

/*****
/* This *.pld file demonstrates the use of basic logic equations in      */
/* describing the I/O behavior of a combinatorial logic circuit.          */
*****/

/** Allowable Target Device--GAL22V10C (DIP) *****/

/** Inputs *****/
Pin 4 = d; /* Dime */
Pin 5 = n; /* Nickel */
Pin 11 = oe2; /* Tri-state output enable */
Pin 10 = !oe1; /* Tri-state output enable (Note negative logic) */
Pin 9 = oe0; /* Tri-state output enable */

/** Outputs *****/
Pin 16 = Q2; /* Nickel and dime */
Pin 15 = Q1; /* Nickel or dime */
Pin 14 = Q0; /* Nickel and not dime */

/** Declarations and Intermediate Variable Definitions--None used *****/

/** State Description for Mealy Machine--None used *****/

/** State Description for Moore Machine--None used *****/

/** Combinatorial I/O Logic Equations and/or Truth Tables *****/
Q2 = n & d; /* Nickel and dime */
Q1 = n # d; /* Nickel or dime */
Q0 = n & !d; /* Nickel and not dime */

/** Logic Equations for Dynamically Configuring PLD *****/
Q2.oe = oe2; /* Tri-state output enable */
Q1.oe = oe1; /* Tri-state output enable */
Q0.oe = oe0; /* Tri-state output enable */

```

```

Name      Demo for a GAL22V10C (DIP);
Partno    Demo_01;
Revision  01;
Date      7/14/98;
Designer  P. David Fisher;
Company    Spartan Embedded Technologies @ Michigan State University;
Assembly  Prototype;
Location   3230 EB;
Device     GAL22V10C (DIP);

/*****
/* This *.pld file demonstrates the use of basic logic equations in
/* describing the I/O behavior of a combinatorial logic circuit.
*****/

/** Allowable Target Device--GAL22V10C (DIP) *****/

Order:
      n, %2, d, %2, oe2, %2, !oe1, %2, oe0, %4, Q2, %2, Q1, %2, Q0 ;

Vectors:

$msg "      !";
$msg "      n d o o o Q Q Q";
$msg "      e e e 2 1 0";
$msg "      2 1 0";
$msg "      ";

      1 1 0 1 0 Z Z Z
      0 0 1 0 1 L L L
      1 0 1 0 1 L H H
      0 1 1 0 1 L H L
      1 1 1 0 1 H H L

```

```

Name          Demo for a GAL22V10C (DIP);
Partno        Demo_02;
Revision      01;
Date          7/14/98;
Designer      P. David Fisher;
Company       Spartan Embedded Technologies @ Michigan State University;
Assembly      Prototype;
Location      3230 EB;
Device        GAL22V10C (DIP);

/*****
/* This *.pld file demonstrates the use of basic logic equations in      */
/* describing the I/O behavior of a combinatorial logic circuit.          */
*****/

/** Allowable Target Device--GAL22V10C (DIP) *****/

/** Inputs *****/
Pin 4 = d; /* Dime */
Pin 5 = n; /* Nickel */
Pin 11 = oe2; /* Tri-state output enable */
Pin 10 = !oe1; /* Tri-state output enable (Note negative logic) */
Pin 9 = oe0; /* Tri-state output enable */

/** Outputs *****/
Pin 16 = Q2; /* Nickel and dime */
Pin 15 = Q1; /* Nickel or dime */
Pin 14 = Q0; /* Nickel and not dime */

/** Declarations and Intermediate Variable Definitions *****/
Field input = [n,d]; /* Declare new variable "input" */
Field output = [Q2..0]; /* Declare new variable "output" */

/** State Description for Mealy Machine--None used *****/

/** State Description for Moore Machine--None used *****/

/** Combinatorial I/O Logic Equations and/or Truth Tables *****/
Table input => output {
0 => 0; 1 => 2; 2 => 3; 3 => 6; /* Default is hex code for numbers */
}

/** Logic Equations for Dynamically Configuring PLD *****/
Q2.oe = oe2; /* Tri-state output enable */
Q1.oe = oe1; /* Tri-state output enable */
Q0.oe = oe0; /* Tri-state output enable */

```

```

Name          Demo for a GAL22V10C (DIP);
Partno        Demo_03;
Revision      01;
Date          7/14/98;
Designer      P. David Fisher;
Company       Spartan Embedded Technologies @ Michigan State University;
Assembly      Prototype;
Location      3230 EB;
Device        GAL22V10C (DIP);

/*****
/* This *.pld file demonstrates the use of basic logic equations in      */
/* describing the I/O behavior of a combinatorial logic circuit.          */
*****/

/** Allowable Target Device--GAL22V10C (DIP) *****/

/** Inputs *****/
Pin 4 = d; /* Dime */
Pin 5 = n; /* Nickel */
Pin 11 = oe2; /* Tri-state output enable */
Pin 10 = !oe1; /* Tri-state output enable (Note negative logic) */
Pin 9 = oe0; /* Tri-state output enable */

/** Outputs *****/
Pin 16 = Q2; /* Nickel and dime */
Pin 15 = Q1; /* Nickel or dime */
Pin 14 = Q0; /* Nickel and not dime */

A = n & d; /* Intermediate variable A */
B = n # d; /* Intermediate variable B */
C = n & !d; /* Intermediate variable C */

/*****
/* Note: These three intermediate variables were not required. They      */
/* were inserted here only to demonstrate the principle. Defining        */
/* them could be used to simplify the logic statements that follow.      */
*****/

/** State Description for Mealy Machine--None used *****/

/** State Description for Moore Machine--None used *****/

/** Combinatorial I/O Logic Equations and/or Truth Tables *****/

Condition {
If A out Q2 out Q1;
If B out Q1;
If C out Q1 out Q0;
}

/** Logic Equations for Dynamically Configuring PLD *****/
Q2.oe = oe2; /* Tri-state output enable */
Q1.oe = oe1; /* Tri-state output enable */
Q0.oe = oe0; /* Tri-state output enable */

```

Fire-Control Demo

Functional Requirements

A state machine must be designed and implemented using the GAL22V10C to satisfy the following requirements:

1. The state machine has four symbolic states: S0, S1, S2, and S3.
2. State S0 is referred to as the “idle state.” When in this state, the output pin “!idle” is asserted and no other outputs are asserted.
3. If while in state S0, the “arm” input pin is asserted and not the “cancel” input pin, then there is a transition to S1 on the next positive edge of the clock.
4. S1 is referred to as an “intermediate state.” No outputs are asserted, and, there is a transition to S2 on the next positive edge of the clock.
5. S2 is referred to as the “armed state.” While in this state, the “ready” output pin is the only output that is asserted.
6. The state machine remains in S2 unless one of two conditions occurs:

- 6.1. If the “cancel” input pin is asserted, the state machine returns to S0.
- 6.2. If the “fire_in” input pin is asserted and not the “cancel” input pin, the state machine goes to S3. S3 is referred to as the “fire state.”
- 7. While in S3, the “ready” and “fire_out” output pins are the only ones that are asserted. After one clock cycle, the state machine unconditionally returns to S0.
- 8. The tri-state logic for all outputs is controlled by a common input pin named “!oe.”
- 9. The asynchronous reset of the internal register is controlled by an input pin named “reset.”
- 10. The synchronous preset of the internal register is controlled by an input pin named “preset.”

Performance Requirement

Our customer wants this fire-control processor to be able to cycle through the four states with a 20 ns maximum elapsed time. Will the GAL22V10C satisfy this requirement? Justify your answer. If it does not, do you have a recommendation for an alternative IC.

```

Name      Demo_04;
Partno    Demo_04_gal22v10;
Revision  01;
Date      7/14/98;
Designer  P. David Fisher;
Company   Spartan Embedded Technologies @ Michigan State University;
Assembly  Prototype;
Location  3230 EB;
Device    GAL22V10C (DIP);

/*****
/* This *.pld file demonstrates how to implement a state machine.      */
*****/

/** Allowable Target Device--GAL22V10C (DIP) *****/

/** Inputs *****/
Pin 1 = clk; /* Clock input */
Pin 2 = reset; /* Asynchronous reset of the internal register */
Pin 3 = preset; /* Synchronous preset of the internal register */
Pin 4 = !oe; /* Tri-state output enable */
Pin 5 = arm; /* Load the ammunition */
Pin 6 = cancel; /* Remove the ammunition and return to idle state */
Pin 7 = fire_in; /* Fire one round and then return to idle state */

/** Outputs *****/
Pin 14 = !idle; /* Control system is in the idle state */
Pin 15 = ready; /* Amunition has been loaded */
Pin 16 = fire_out; /* Fire one round of ammunition */
Pin 17 = A; /* State variable */
Pin 18 = B; /* State variable */

/** Declarations and Intermediate Variables *****/
Field state = [A,B]; /* The state vector has two bits */
$Define S0 'b'00 /*The idle state */
$Define S1 'b'01 /*The intermediate state */
$Define S2 'b'10 /*The ready state */
$Define S3 'b'11 /*The fire state */

/** State Description for Mealy Machine--None used *****/

/** State Description for Moore Machine *****/

Sequence state {

Present S0
    If cancel next S0; /* Stay in idle state */
    If !cancel & arm next S1; /* Go to intermediate state */
    Out idle; /* Idle output is asserted */

Present S1
    Next S2; /* Unconditionally go to the ready state */

Present S2
    If cancel next S0; /* Cancel mission and return to idle state */
    If !cancel & !fire_in next S2; /* Wait for next command */
    If !cancel & fire_in next S3; /* Fire the ammunition */
    Out ready; /* Ready output is asserted */

```

Present S3

```
    Next S0; /* Unconditionally return to the idle state */
    Out ready Out fire_out; /* Ready & fire_out outputs asserted */
}
```

```
/** Combinatorial I/O Logic Equations and/or Truth Tables--not used **/
```

```
/** Logic Equations for Dynamically Configuring PLD *****/
idle.oe = oe; /* Tri-state output enable */
ready.oe = oe; /* Tri-state output enable */
fire_out.oe = oe; /* Tri-state output enable */
A.oe = oe; /* Tri-state output enable */
B.oe = oe; /* Tri-state output enable */
state.ar = reset; /* Asynchronous reset of register */
state.sp = preset; /* Synchronous preset of register */
```

```

Name      Demo_04;
Partno    Demo_04_gal22v10;
Revision  01;
Date      7/14/98;
Designer  P. David Fisher;
Company   Spartan Embedded Technologies @ Michigan State University;
Assembly  Prototype;
Location  3230 EB;
Device    GAL22V10C (DIP);

```

```

/*****
/* This *.pld file demonstrates how to implement a state machine.      */
*****/

```

```

/** Allowable Target Device--GAL22V10C (DIP) *****/

```

Order:

```

    clk, %2, reset, %2, preset, %2, !oe, %2, arm, %2, cancel, %2,
    fire_in, %4, A, %2, B, %2, !idle, %2, ready, %2, fire_out;

```

Vectors:

```

$msg "      c  r  p  !  a  c  f      A  B  !  r  f  ";
$msg "      l  e  r  o  r  a  i      i  e  i  ";
$msg "      k  s  e  e  m  n  r      d  a  r  ";
$msg "      e  s      c  e      l  d  e  ";
$msg "      t  e      e  _  e  y  _  ";
$msg "      t      l  _  o  ";
$msg "      n      n  u  ";
$msg "      t  ";

    0  0  0  1  0  0  0      Z  Z  Z  Z  Z
    0  1  0  0  0  0  0      L  L  L  L  L
    C  0  1  0  0  0  0      H  H  H  H  H
    0  1  0  0  0  0  0      L  L  L  L  L
    C  0  0  0  0  0  0      L  L  L  L  L
    C  0  0  0  1  0  0      L  H  H  L  L
    C  0  0  0  X  0  X      H  L  H  H  L
    C  0  0  0  X  1  X      L  L  L  L  L
    C  0  0  0  1  0  X      L  H  H  L  L
    C  0  0  0  X  0  X      H  L  H  H  L
    C  0  0  0  X  0  0      H  L  H  H  L
    C  0  0  0  X  0  1      H  H  H  H  H
    C  0  0  0  X  0  X      L  L  L  L  L
    C  0  1  0  0  0  0      H  H  H  H  H
    0  1  0  0  0  0      L  L  L  L  L

```